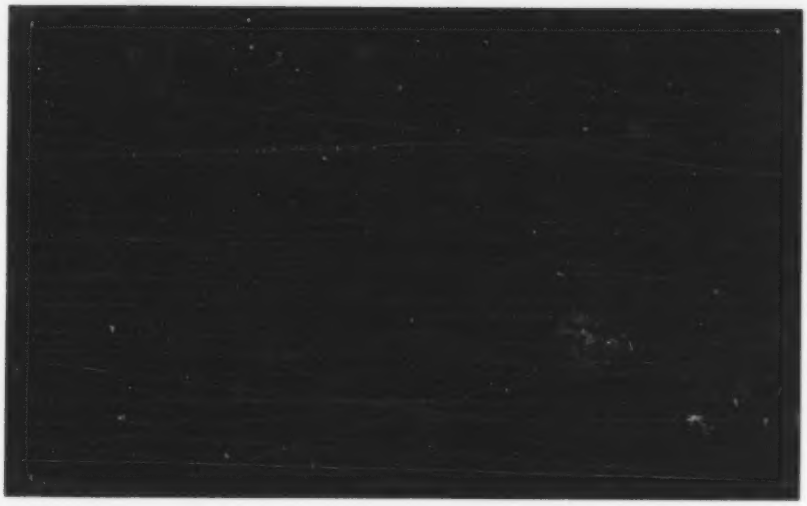
**CENTRE DE RECHERCHE**
**SUR LES TRANSPORTS**
(C.R.T.)
*CENTRE FOR RESEARCH*
*ON TRANSPORTATION*

UNIVERSITÉ DE MONTRÉAL
ÉCOLE DES HAUTES ÉTUDES COMMERCIALES
ÉCOLE POLYTECHNIQUE



CENTRE DE RECHERCHE SUR LES TRANSPORTS
UNIVERSITÉ DE MONTRÉAL
C.P. 6128, SUCCURSALE A,
MONTRÉAL
CANADA H3C 3J7

TÉLÉPHONE: (514) 343-7575
TÉLEX: 05-24146 BIBPOLYTEC
EMAIL: crt@crt.umontreal.ca
TÉLÉCOPIEUR (FAX): (514) 343-7121

CRT-708

# NEW INSERTION AND POST-OPTIMIZATION PROCEDURES FOR THE TRAVELING SALESMAN PROBLEM

by

Michel Gendreau[1]
Alain Hertz[2,3]
and
Gilbert Laporte[1,3]

[1] Centre de recherche sur les transports, Université de Montréal, C.P. 6128 succursale A, Montréal, Québec, Canada H3C 3J7.

[2] Département d'Informatique et de Recherche opérationnelle, Université de Montréal, C.P. 6128, succursale A, Montréal, Québec, Canada H3C 3J7.

[3] GERAD, École des Hautes Études Commerciales de Montréal, 5255 avenue Decelles, Montréal, Québec, Canada H3T 1V6.

Centre de recherche sur les transports - Publication #708

August 1990

Dépôt légal - 3e trimestre 1990

91-00317/


## ABSTRACT

This paper describes a new insertion procedure and a new post-optimization routine for the traveling salesman problem. The combination of the two methods results in an efficient algorithm (GENIUS) which outperforms known alternative heuristics in terms of solution quality and computing time.


Key words : traveling salesman problem, insertion heuristics.

- - - - - - - - -


## RÉSUMÉ

On décrit dans cet article une nouvelle procédure d'insertion ainsi qu'une nouvelle méthode de post-optimisation pour le problème du voyageur de commerce. La combinaison de ces deux méthodes donne lieu à un algorithme efficace (GENIUS) qui surpasse d'autres heuristiques connues, à la fois en termes de qualité de solution et de temps de calcul.


Mots-clés : problème du voyageur de commerce, heuristiques d'insertion.

The Traveling Salesman Problem (TSP) is one of the most widely studied problems in combinatorial optimization. It can be defined as follows. Let $G = (V, A)$ be a graph where $V = \{v_1, \ldots, v_n\}$ is the vertex set and where $A = \{(v_i, v_j) : v_i, v_j \in V\}$ is the arc set. With each arc $(v_i, v_j)$ is associated a non-negative cost $c_{ij}$. The TSP consists in finding the least cost circuit passing through every vertex exactly once. Such a circuit is said to be Hamiltonian. The problem is symmetric if and only if $c_{ij} = c_{ji}$ for all i,j. It satisfies the triangle inequality if and only if $c_{ij} + c_{jk} \geq c_{ik}$ for all i, j, k; a particular case is that of Euclidean problems whose costs correspond to Euclidean distances.

The most common interpretation of the TSP is that of a salesman seeking to determine a shortest Hamiltonian circuit through n cities. A number of Vehicle Routing Problems can also be derived from the TSP. One of the vertices is then designated as a depot and a number of least cost vehicle routes must be designed from the depot to the remaining vertices, subject to side constraints. It is often possible to solve VRPs by adapting TSP algorithms (e.g. Laporte, Nobert and Desrochers 1985; Laporte, Mercure and Nobert 1986). In recent years, new TSP applications have been described in several other fields such as drilling of printed circuit boards (Reinelt 1989; Grötschel, Jünger and Reinelt 1989), sequencing of punch operations in a flexible manufacturing cell (Chauny et al. 1987) and crystallography (Bland and Shallcross 1989).

Several of these applications are of large-scale and can rarely be tackled by means of exact algorithms. Some notable exceptions are the drilling problems solved by Padberg and Rinaldi (1990), using a polytopal cutting planes algorithm. Most of these problems require fast low complexity heuristics for their solution. The object of this paper is to describe such a heuristic which outperforms previously published methods on test problems. It includes an improved insertion procedure and a new post-optimization routine which can also be used in conjunction with other algorithms.

The remainder of this paper is organized as follows. In Section 1, we briefly review a number of TSP heuristics relevant to this paper. The insertion procedure (GENI) and the improvement method (US) are described in the next two sections. Computational results, reported in Section 4, indicate that the combination of these two heuristics (GENIUS) compares favourably with other existing algorithms. The conclusion follows in Section 5.

## 1. Heuristic Algorithms for the TSP

Broadly speaking, TSP heuristics can be classified into tour construction procedures which consist of gradually building a solution by adding a new vertex at each step, and tour improvement procedures which improve upon a feasible solution by performing various exchanges. The best methods are two-phase procedures that combine these two features.

Tour construction methods include the well-known nearest neighbour heuristic (see Rosenkrantz, Stearns and Lewis 1977) and a number of insertion procedures that use various criteria (Rosenkrantz, Stearns and Lewis 1977; Stewart 1977; Norback and Love 1977, 1979; Or 1976). For empirical comparisons and asymptotic analyses of these heuristics, see Golden and Stewart (1985) as well as Ong and Huang (1989). Some authors have also proposed linear or near-linear time heuristics applicable to large-scale TSPs arising from manufacturing contexts. These include partitioning algorithms (Karp 1977), strip heuristics (Supowit, Reingold and Plaisted 1983; Daganzo 1984), the Delaunay partition scheme described by Reinelt (1989) and space-filling curve approaches implemented by Chauny et al. (1987) and by Bartholdi and Platzman (1988). These methods are meant to quickly provide good solutions, often in real-time. However, as far as solution quality is concerned, they are usually not the most refined algorithms available.

Classical improvement procedures are those of Lin (1965), Lin and Kernighan (1973) and Or (1976). The latter method (Or-Opt) seems particularly successful and has been used by a number of authors. It can be summarized as follows:

**Step 1.** Consider an initial tour and set t:=1 and s:=3.

**Step 2.** Remove from the tour a string of s consecutive vertices, starting with the vertex in position t, and tentatively insert it between all remaining pairs of consecutive vertices of the tour.

- If the tentative insertion decreases the cost of the tour, implement it immediately, thus defining a new tour; set t:=1 and repeat Step 2.
- If no tentative insertion decreases the cost of the tour, set t:=t+1. If t=n+1, then proceed to Step 3; otherwise repeat Step 2.

**Step 3.** Set t:=1 and s:=s-1. If s>1, then go to Step 2; otherwise stop. ∎

As shown by Stewart (1987), such algorithms can often be accelerated by limiting the number of exchange candidates. In addition, a number of global optimization techniques have also been adapted for the TSP. The first, Simulated Annealing, is derived from the early work of Metropolis et al. (1953) and was first applied to the field of combinatorial optimization by Kirkpatrick, Gelatt and Vecchi (1983). Its application to the TSP by Bonomi and Lutton (1984), Rossier, Troyon and Liebling (1986), Golden and Skiscim (1986) and Troyon (1988) seems to have produced mixed results. On this subject, see the recent analysis made by Soriano (1989). The second of these methods, Tabu Search, is due to Glover (1977) and its application to the TSP seems more hopeful. Recent implementations are those of Knox (1988), Malek (1988) and Fiechter (1990). Both Simulated Annealing and Tabu Search can be termed as open-ended improvement methods. They do not appear to be the most appropriate techniques for quickly generating good tours and their performance is directly related to running time.

One of the better two-phase methods is the so-called CCAO heuristic of Golden and Stewart (1985). It was designed for (planar) symmetric Euclidean problems and it exploits their geometrical properties. This algorithm executes in sequence a tour construction phase and an improvement step. Here, we follow the description given by its authors:

**Step 1.** Define an initial (partial) tour by forming the convex hull of vertices.

**Step 2.** For each vertex $v_k$ not yet contained in the tour, identify the two adjacent vertices $v_{i_k}$ and $v_{j_k}$ on the tour such that $c_{i_k k} + c_{k j_k} - c_{i_k j_k}$ is minimized.

**Step 3.** Select the vertex $v_{k*}$ that maximizes the angle between arcs $(v_{i_k}, v_k)$ and $(v_k, v_{j_k})$ in the tour and insert it between $v_{i_{k*}}$ and $v_{j_{k*}}$.

**Step 4.** Repeat Steps 2 and 3 until a Hamiltonian tour through all vertices is obtained.

**Step 5.** Apply the Or-Opt procedure to the tour and stop. ∎

On fourteen out of fifteen problems tested by Golden and Stewart, the CCAO algorithm yielded a solution having a cost within 2% of the best known cost (in five cases, all problems with n=100, the best available solution was also known to be optimal). Yet, in spite of their good performance, two-phase methods such as CCAO suffer from two serious drawbacks. First, several unpromising moves are considered: this occurs whenever an attempt is made to insert $v_k$ between two vertices $v_{i_k}$ and $v_{j_k}$ distant from $v_k$. Second, the insertion phase of these methods is myopic in the following sense. Since insertions are executed sequentially without much concern for global optimality, they may result in a succession of bad decisions that the post-optimization phase will be unable to undo. The algorithm we are proposing attempts fewer insertions but executes each one more carefully, by performing a limited number of local transformations of the tour, simultaneously with the insertion itself.

## 2. A Generalized Insertion Procedure (GENI)

The principles behind our Generalized Insertion Procedure (GENI) apply equally well to symmetric and asymmetric problems. However, in order to simplify the presentation, and since our computational results are from symmetric problems only, we have chosen to restrict our description to this case. Extending GENI to asymmetric problems is straightforward.

The main feature of GENI is that insertion of a vertex v in the tour does not necessarily take place between two vertices which are consecutive when they are first considered. However, after insertion, these two vertices become adjacent to v. Suppose we wish to insert v between any two vertices $v_i$ and $v_j$ of the tour. Consider a given orientation of the tour, $v_k$ a vertex on the path from $v_j$ to $v_i$ and $v_\ell$, a vertex on the path from $v_i$ to $v_j$. For any vertex $v_h$ on the tour, let $v_{h-1}$ be its predecessor and $v_{h+1}$ its successor. Insertion of v between $v_i$ and $v_j$ can be done in one of two ways, as illustrated in Figures 1 and 2.

Type I Insertion. Here $v_k \neq v_i$ and $v_k \neq v_j$. Inserting v in the tour results in the deletion of arcs $(v_i, v_{i+1})$, $(v_j, v_{j+1})$ and $(v_k, v_{k+1})$ and in their replacement by $(v_i, v)$, $(v, v_j)$, $(v_{i+1}, v_k)$ and $(v_{j+1}, v_{k+1})$. This implies that the two paths $(v_{i+1}, \ldots, v_j)$ and $(v_{j+1}, \ldots, v_k)$ are reversed. Also note that setting j=i+1 and k=j+1 yields the standard insertion procedure.

Type II Insertion. Here $v_k \neq v_j$ and $v_k \neq v_{j+1}$; $v_\ell \neq v_i$ and $v_\ell \neq v_{i+1}$. Inserting v in the tour results in the deletion of $(v_i, v_{i+1})$, $(v_{\ell-1}, v_\ell)$, $(v_j, v_{j+1})$ and $(v_{k-1}, v_k)$. These arcs are replaced by $(v_i, v)$, $(v, v_j)$, $(v_\ell, v_{j+1})$, $(v_{k-1}, v_{\ell-1})$ and $(v_{i+1}, v_k)$. As above, the paths $(v_{i+1}, \ldots, v_{\ell-1})$ and $(v_\ell, \ldots, v_j)$ are reversed.

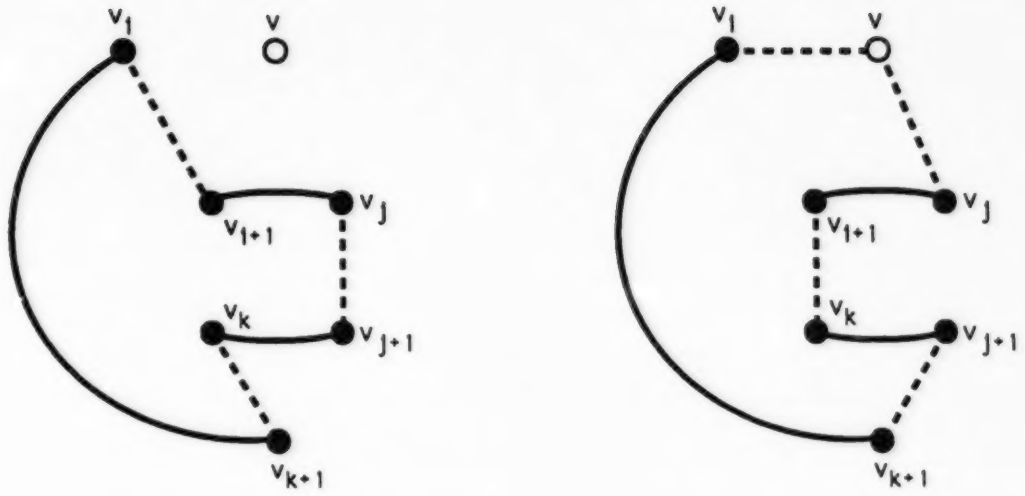



Figure 1 : Type I insertion of vertex v between $v_i$ and $v_j$ .

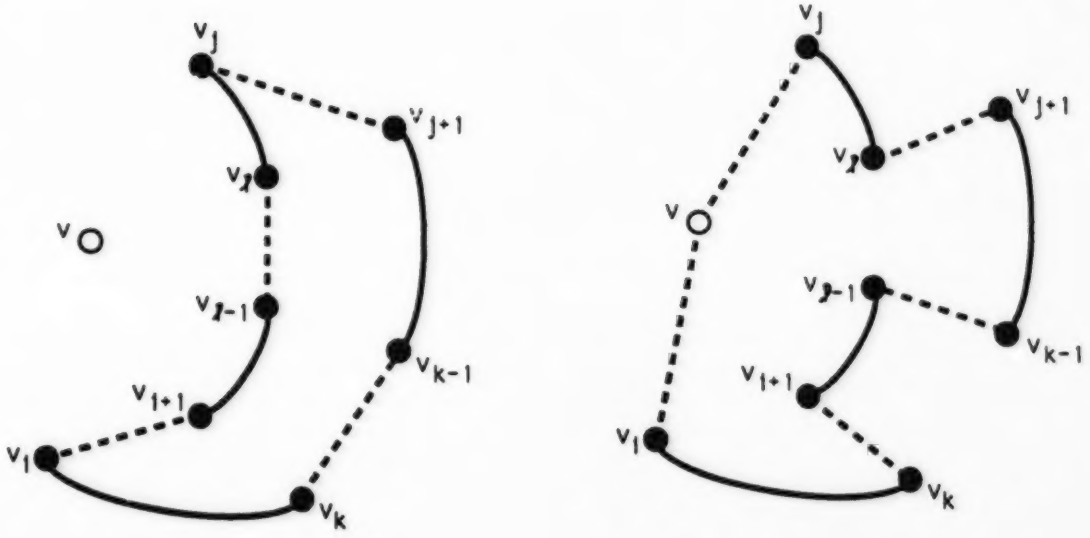



Figure 2 : Type II insertion of vertex v between $v_i$ and $v_j$.

The GENI algorithm considers for each possible insertion, the two possible orientations of the tour. Since the potential number of choices for $v_i$, $v_j$, $v_k$, $v_\ell$ is of the order of $n^4$, we limit the search as follows. For any vertex $v \in V$, define its p-neighbourhood $N_p(v)$ as the set of its p closest neighbours on the tour, excluding v itself; if v has fewer than p neighbours on the tour, they all belong to $N_p(v)$. Then, for a given parameter p, we first select $v_i$, $v_j \in N_p(v)$, $v_k \in N_p(v_{i+1})$ and $v_\ell \in N_p(v_{j+1})$. We also consider all insertions of v between two consecutive vertices $v_i$ and $v_{i+1}$, as long as $v_i \in N_p(v)$. In practice, p is a relatively small number.

The GENI algorithm can now be described:

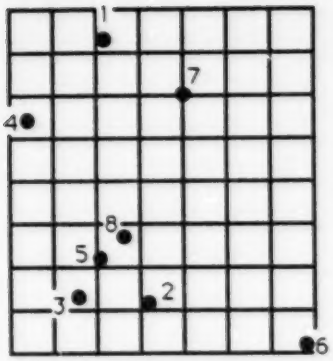
<u>Step 1</u>. Create an initial tour by selecting an arbitrary subset of three vertices. Initialize the p-neighbourhoods of all vertices.

<u>Step 2</u>. Arbitrarily select a vertex v not yet on the tour. Implement the least cost insertion of v considering the two possible orientations of the tour and the two insertion types. Update the p-neighbourhoods of all vertices to account for the fact that v is now on the tour.

<u>Step 3</u>. If all vertices are now part of the tour, stop. Otherwise go back to Step 2. ∎

The complexity of this procedure is determined as follows. In Step 2, $O(p^4)$ choices of $v_i$, $v_j$, $v_k$ and $v_\ell$ must be considered. For the best choice, v is inserted in the tour which is then updated: this requires O(n) time. Since Step 2 is executed n-3 times, the overall complexity of GENI is $O(np^4 + n^2)$.

| vertices | coordinates x | y |
|---|---|---|
| 1 | 21.8 | 72.8 |
| 2 | 31.5 | 10.8 |
| 3 | 16.3 | 13.7 |
| 4 | 3.7 | 54.8 |
| 5 | 21.0 | 21.7 |
| 6 | 69.3 | 1.3 |
| 7 | 40.8 | 60.1 |
| 8 | 27.8 | 26.2 |



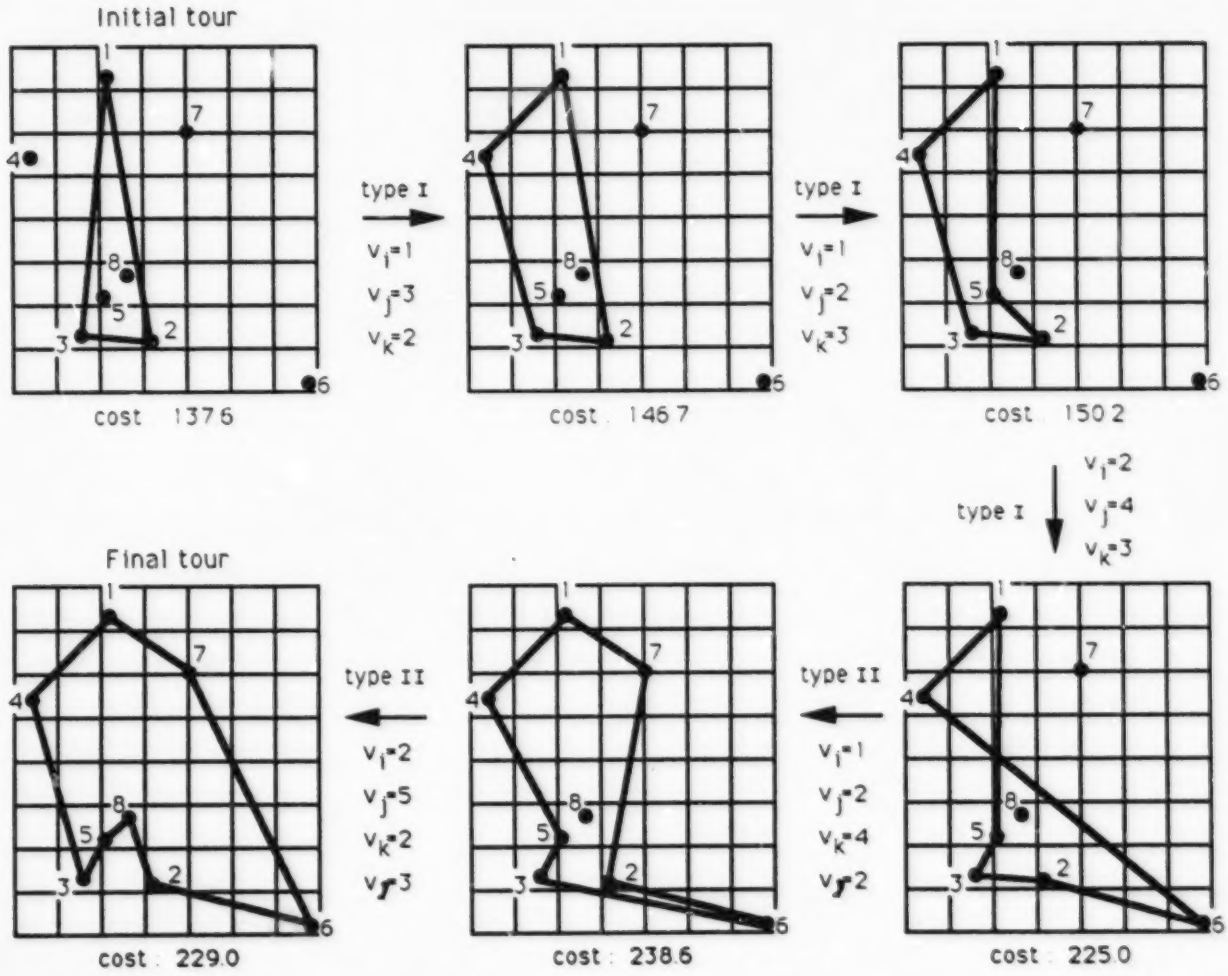



Figure 3 : Illustration of the GENI algorithm on an 8-vertex problem.

It is worth noting that even in Euclidean problems, inserting a new vertex in the tour sometimes results in a lesser cost. This never happens in standard insertion procedures. To illustrate, consider the example depicted in Figure 3. Here p=4 and vertices are inserted in the tour in numerical order. One interesting phenomenon occurs when vertex 6 is inserted: the procedure then produces a tour which crosses itself. This is so because the best available insertion, between vertices 1 and 2, is infeasible since vertex 1 does not belong to $N_4(6)$. Also, a cost reduction from 238.6 to 229.0, occurs when vertex 8 is inserted. Note that the final solution is not optimal. Indeed, the best tour (1, 7, 6, 2, 3, 5, 8, 4, 1) has a cost of 223.0 and can be reached with p=5.

## 3. Unstringing and Stringing (US)

In addition to GENI, we have also developed a post-optimization procedure which consists of removing a vertex from a feasible tour and of inserting it back. By analogy with bead stringing, these processes are called "Unstringing" and "Stringing" (US). This procedure can be applied to tours produced by any algorithm. The Stringing procedure is identical to Step 2 of GENI. The Unstringing procedure is just the reverse. When a vertex $v_i$ is removed, we consider two ways of reconnecting the tour. These are illustrated in Figures 4 and 5.

**Type I Unstringing.** Let $v_j \in N_p(v_{i+1})$ and, for a given orientation of the tour, let $v_k \in N_p(v_{i-1})$ be a vertex on the path $(v_{i+1}, \ldots, v_{j-1})$. Then arcs $(v_{i-1}, v_i)$, $(v_i, v_{i+1})$, $(v_k, v_{k+1})$ and $(v_j, v_{j+1})$ are deleted, while arcs $(v_{i-1}, v_k)$, $(v_{i+1}, v_j)$ and $(v_{k+1}, v_{j+1})$ are inserted. Also, the two paths $(v_{i+1}, \ldots, v_k)$ and $(v_{k+1}, \ldots, v_j)$ are reversed.

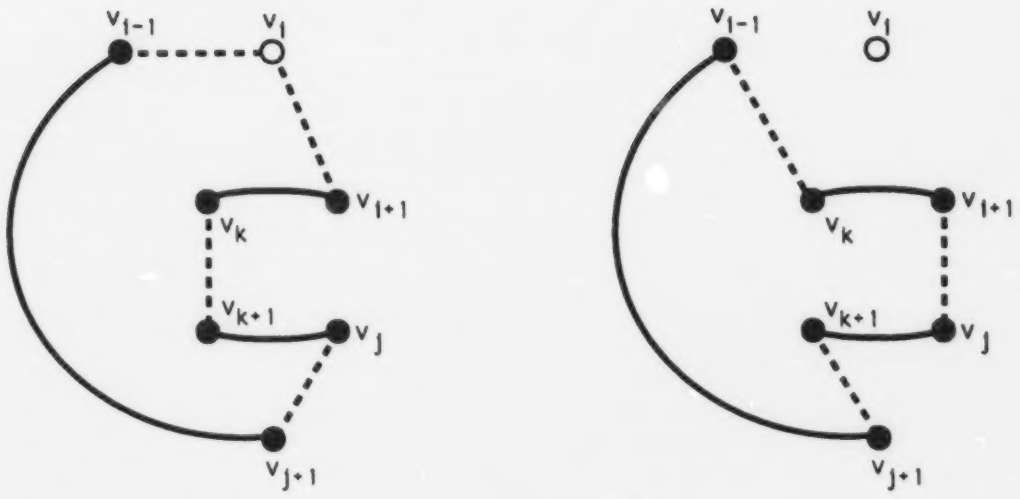



Figure 4 : Type I Unstringing of vertex $v_i$ from the tour.

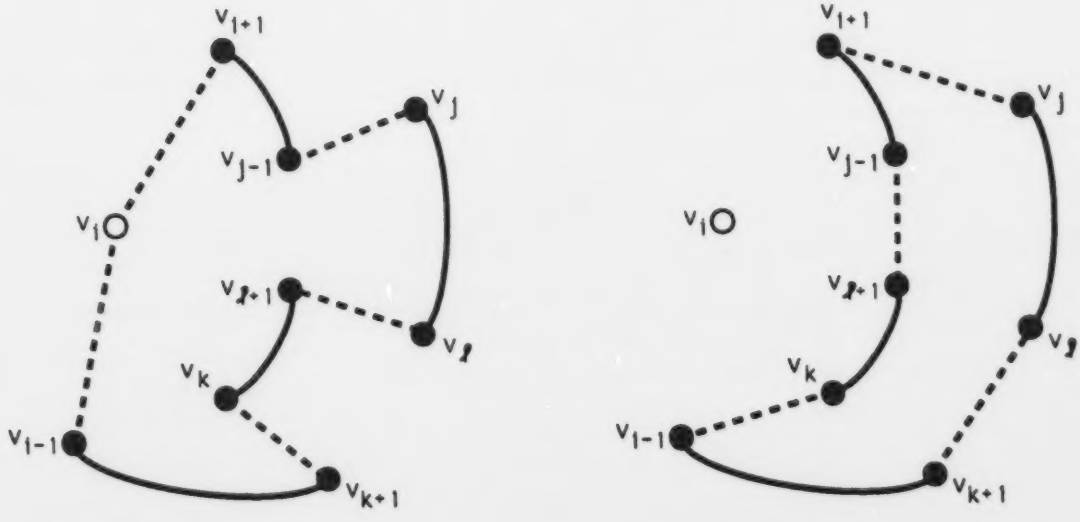



Figure 5 : Type II Unstringing of vertex $v_i$ from the tour.

**Type II Unstringing.** Let $v_j \in N_p(v_{i+1})$. For a given orientation of the tour, let $v_k \in N_p(v_{i-1})$ be a vertex on the path $(v_{j+1}, \ldots, v_{i-2})$; let also $v_\ell \in N_p(v_{k+1})$ be a vertex on the path $(v_j, \ldots, v_{k-1})$. Then arcs $(v_{i-1}, v_i)$, $(v_i, v_{i+1})$, $(v_{j-1}, v_j)$, $(v_\ell, v_{\ell+1})$ and $(v_k, v_{k+1})$ are deleted while arcs $(v_{i-1}, v_k)$, $(v_{\ell+1}, v_{j-1})$, $(v_{i+1}, v_j)$ and $(v_\ell, v_{k+1})$ are inserted. Also, the two paths $(v_{i+1}, \ldots, v_{j-1})$ and $(v_{\ell+1}, \ldots, v_k)$ are reversed.

The US procedure can now be described:

**Step 1.** Consider an initial tour $\tau$ of cost $z$. Set $\tau^*:=\tau$, $z^*:=z$ and $t:=1$.

**Step 2.** Starting from tour $\tau$, apply the Unstringing and Stringing procedures with vertex $v_t$, considering in each case the two possible types of operation and the two tour orientations. Let $\tau'$ be the tour obtained and let $z'$ be its cost. Set $\tau:=\tau'$ and $z:=z'$.

- If $z < z^*$, set $\tau^*:=\tau$, $z^*:=z$ and $t:=1$; repeat Step 2.
- If $z > z^*$, set $t:=t+1$.
- If $t=n+1$, stop: the best available tour is $\tau^*$ and its cost is equal to $z^*$. Otherwise, repeat Step 2. ∎

Note that in the spirit of Simulated Annealing and Tabu Search methods, Step 2 is not necessarily applied to the best tour available. In other words, the costs of two consecutive tours produced by Step 2 may increase. However, the best known tour is always stored. This rule prevents the procedure from being trapped in a local optimum. It has been compared with alternative rules and found to be superior on most test problems.

## 4. Computational Results

In order to assess the efficiency of GENI and US, and to make comparisons with existing methods, we have carried out several series of computational tests both on randomly generated problems and on TSPs described in the O.R. literature. The first set of problems were derived by first generating $n$ points in $[0,100]^2$, according to a continuous uniform distribution, and by then computing the symmetric Euclidean distances (costs) between these points.

The various procedures used in the computational tests are:

- C : the Convex Hull procedure in Step 1 of the CCAO algorithm;
- CA : the insertion phase (Steps 2, 3 and 4) of CCAO;
- O : the Or-Opt algorithm;
- GENI: the proposed generalized insertion algorithm and
- US : the Unstringing and Stringing routine.

Several combinations of these procedures were tested: CCA, CCAO, GENI, GENIO and GENIUS. In addition, we have tested a modified version of GENI (CGENI) which consists of using the Convex Hull of vertices as an initial tour. The various versions of GENI were tested for $p=2, \ldots, 7$. The Convex Hull procedure (C) is the $O(n \ln n)$ GRAHAMHULL routine without polar coordinates, described in Preparata and Shamos (1985), pp. 108-109. For CCA and O, we programmed a careful and efficient implementation of the algorithms, as described in Section 1. All codes were written in Pascal and tests were executed on a SUN SPARC Work Station. The first set of computational results are reported in Table 1. For the various algorithms that were compared, we provide the average solution costs (in column "COST") and the average computation times in seconds (in column "TIME") over 100 independently generated problems.

Table 1: Computational results for random Euclidean problems.

| | p | n=100 | | n=200 | | n=300 | | n=400 | | n=500 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | COST | TIME | COST | TIME | COST | TIME | COST | TIME | COST | TIME |
| GENI | 2 | 842.5 | 0.4 | 1176.3 | 1.4 | 1431.2 | 3.1 | 1644.9 | 5.4 | 1838.8 | 8.4 |
| | 3 | 815.2 | 0.5 | 1134.1 | 1.7 | 1382.1 | 3.5 | 1583.2 | 6.0 | 1770.4 | 9.2 |
| | 4 | 807.3 | 0.8 | 1121.6 | 2.3 | 1364.8 | 4.5 | 1566.6 | 7.4 | 1748.5 | 11.0 |
| | 5 | 803.7 | 1.4 | 1115.3 | 3.6 | 1355.3 | 6.6 | 1557.7 | 10.1 | 1737.7 | 14.4 |
| | 6 | 799.1 | 2.6 | 1111.1 | 6.1 | 1351.8 | 10.4 | 1549.6 | 15.2 | 1728.8 | 20.9 |
| | 7 | 798.1 | 4.5 | 1109.1 | 10.2 | 1348.7 | 16.7 | 1547.5 | 23.8 | 1724.9 | 31.8 |
| CGENI | 2 | 842.3 | 0.4 | 1183.1 | 1.5 | 1436.0 | 3.2 | 1646.5 | 5.6 | 1839.4 | 8.6 |
| | 3 | 813.7 | 0.5 | 1134.6 | 1.7 | 1383.6 | 3.6 | 1585.4 | 6.1 | 1768.0 | 9.3 |
| | 4 | 804.6 | 0.8 | 1121.6 | 2.3 | 1365.4 | 4.6 | 1565.2 | 7.5 | 1746.2 | 11.1 |
| | 5 | 800.8 | 1.4 | 1114.7 | 3.6 | 1354.8 | 6.6 | 1556.2 | 10.2 | 1734.1 | 14.5 |
| | 6 | 799.4 | 2.5 | 1110.0 | 6.0 | 1351.6 | 10.2 | 1549.8 | 15.1 | 1727.4 | 20.8 |
| | 7 | 797.5 | 4.4 | 1107.7 | 10.0 | 1348.1 | 16.4 | 1546.9 | 23.5 | 1724.9 | 31.3 |
| GENIO | 2 | 809.5 | 2.5 | 1126.7 | 16.5 | 1369.9 | 52.0 | 1576.4 | 117.9 | 1756.5 | 200.9 |
| | 3 | 803.2 | 1.8 | 1117.6 | 9.9 | 1358.0 | 30.7 | 1556.3 | 65.9 | 1739.6 | 106.8 |
| | 4 | 799.8 | 1.8 | 1111.7 | 8.7 | 1352.2 | 23.2 | 1550.5 | 50.5 | 1730.6 | 79.9 |
| | 5 | 799.4 | 2.3 | 1108.0 | 8.9 | 1347.0 | 21.8 | 1547.2 | 43.4 | 1725.1 | 67.4 |
| | 6 | 796.1 | 3.4 | 1105.0 | 11.0 | 1345.4 | 23.6 | 1542.2 | 43.2 | 1719.4 | 65.8 |
| | 7 | 794.9 | 5.4 | 1104.0 | 14.6 | 1343.1 | 29.5 | 1540.6 | 50.3 | 1716.5 | 72.8 |
| GENIUS | 2 | 812.0 | 2.1 | 1130.7 | 9.4 | 1379.9 | 22.2 | 1582.9 | 43.2 | 1769.6 | 67.2 |
| | 3 | 796.8 | 2.5 | **1107.6** | **10.0** | **1347.0** | **23.5** | **1545.4** | **40.8** | **1725.0** | **70.0** |
| | 4 | 791.8 | 3.4 | **1100.1** | **11.8** | **1337.1** | **26.2** | **1535.5** | **46.0** | **1710.7** | **71.1** |
| | 5 | 791.2 | 5.4 | 1096.7 | 15.9 | **1332.5** | **33.3** | **1531.9** | **54.2** | **1704.3** | **82.4** |
| | 6 | 788.3 | 8.7 | 1095.3 | 23.8 | **1332.2** | **45.4** | **1528.1** | **70.9** | **1701.0** | **109.1** |
| | 7 | 788.6 | 13.6 | 1095.6 | 35.6 | 1330.6 | 66.7 | **1526.9** | **107.2** | **1698.6** | **142.4** |
| CCA | | 820.0 | 0.7 | 1149.4 | 3.5 | 1405.6 | 8.6 | 1619.1 | 16.2 | 1810.9 | 27.3 |
| CCAO | | 798.1 | 2.1 | 1111.0 | 14.9 | 1354.8 | 48.5 | 1556.8 | 112.1 | 1735.3 | 196.6 |

A number of observations can be derived from these results. First, comparing GENI and CGENI shows that these two procedures are more or less equivalent to one another, both in terms of the objective function and of computing speed. In other words, whether the generalized insertion routine is initialized from scratch or from the convex hull is immaterial. Similar comparisons were made between GENIO and CGENIO as well as between GENIUS and CGENIUS and in both cases the same conclusion was reached. Not surprisingly, for GENI, GENIO and GENIUS, solution quality usually improves as p becomes larger. While for GENI and GENIUS, computing times almost always increase with p, this is not the case for GENIO: here they attain their minimum at an intermediate value of p. Our proposed post-optimization procedure (US) is better than Or-Opt (O), when executed after GENI. This can be seen by comparing GENIO and GENIUS: when $p>3$, the latter algorithm always yields better results than GENIO, although it is slower for $p>5$. Similarly, comparing the two procedures that do not use post-optimization (GENI and CCA) shows that GENI is always better than CCA provided $p>3$, and quicker when p is not too large. Also, when $n>200$, using our generalized insertion procedure without post-optimization (GENI) with a large enough value of p appears to be faster and better than CCAO.

Overall, the algorithm that yields the best solutions is GENIUS with $p>4$. When $n>200$, there always exists a range of values of p for which GENIUS is at the same time better and faster than CCAO; the corresponding results are indicated in bold characters in Table 1. For the values of n considered, using GENIUS with p in that range seems a good choice. As the relative advantage of GENI and GENIUS over CCAO tends to increase with problem size, larger values of p should probably be selected for larger n.

In addition to these tests on random problems, we have compared GENI and GENIUS with other algorithms on the following problems on which alternative heuristics had already been tested and for which optimal values were available:

-K24 : a 100-vertex problem (number 24) described in Krolak, Felts and Marble (1971);

-GRID100: a 100-vertex grid problem generated as in Cerny (1985);
-GRID400: a 400-vertex problem generated in the same fashion;
-G442 : the Grötschel, Jünger and Reinelt (1989) 442-vertex problem and
-P532 : the Padberg and Rinaldi (1987) 532-vertex problem.

All problems were solved with GENI, GENIUS, CCA and CCAO. For these, computation times are for the SUN SPARC Work Station. We report solution values and computing times for these problems, when available, obtained with Lin's 2-opt algorithm (Reinelt 1989), with the Lin and Kerninghan algorithm (Reinelt 1989), with Simulated Annealing (Rossier, Troyon and Liebling 1986; Troyon 1988), and with Tabu Search (Fiechter 1990). These results are reported in Table 2, along with the various computers used. In addition to the COST and TIME columns, we provide the values of p used in GENI and GENIUS.

Our results indicate that on GRID100 and GRID400, GENI (and thus GENIUS) finds an optimal solution. GENIUS also finds the optimum for K24. For the last two problems, GENI generates better solutions than all reported heuristics, except Simulated Annealing and Tabu Search. GENIUS is only surpassed by Tabu Search on a single problem (P532), with respect to solution cost, but would appear to be much faster given the relative speeds of the two computers involved.

## 5. CONCLUSION

We have described a new insertion heuristic and a new post-optimization procedure for the TSP. Computational experiments carried out both on randomly generated problems and on problems described in the O.R. literature confirm the relative efficiency of the two proposed methods. The GENIUS algorithm derived from a combination of the two methods compares advantageously with the best alternative heuristic approaches for the TSP, both in terms of solution quality and in terms of computation times. The proposed routines can be used in conjunction with other heuristics and can easily be extended to a number of Vehicle Routing Problems and to other types of combinatorial optimization problems in which a minimum cost permutation of n objects must be determined.

Table 2 : Computational results for problems described in the O.R. literature.

| | PROBLEM | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | K24 | | | GRID100 | | | GRID400 | | | G442 | | | P532 | | |
| | p | COST | TIME | p | COST | TIME | p | COST | TIME | p | COST | TIME | p | COST | TIME |
| GENI | 5 | 21792 | 1.7 | 3 | 100 | 0.6 | 3 | 400 | 7.5 | 5 | 52.77 | 14.0 | 5 | 28433 | 18.8 |
| GENIUS | 5 | 21282 | 4.7 | 3 | 100 | 2.6 | 3 | 400 | 23.5 | 6 | 51.27 | 114.3 | 5 | 28175 | 95.4 |
| CCA | | 21512 | 0.8 | | 106.2 | 0.5 | | 471.6 | 11.0 | | 55.25 | 23.2 | | 29559 | 35.7 |
| CCAO | | 21320 | 2.0 | | 102.5 | 1.4 | | 419.1 | 151.8 | | 52.79 | 117.7 | | 28726 | 277.4 |
| 2-OPT | | | | | 103.3 | 18.0[1] | | | | | 54.67 | 29.6[1] | | | |
| LIN-KERNIGHAN | | | | | | | | | | | 52.54 | 46.1[1] | | | |
| SIMULATED ANNEALING | | | | | 100 | 22.3[2] | | | | | 51.42 | 238.0[3] | | 28418[4] | 448.0[3] |
| TABU SEARCH | | | | | | | | | | | 51.45[5]<br>51.10[6]<br>51.94[7] | 254.0[8] | | 28090[5]<br>27839[6]<br>28422[7] | 267.0[8] |
| OPTIMAL VALUE | | 21282[9] | | | 100[10] | | | 400[10] | | | 50.67[11] | | | 27686[12] | |

1 Reinelt (1989). SUN 3/60 Work Station.
2 Rossier, Troyon and Liebling (1986). CYBER 170-835.
3 Troyon (1988). VAX 8600.
4 P532 problem with truncated distances: the optimal solution value for the truncated version is 27424.
5 Fiechter (1990). Average solution value over 100 trials.
6 Fiechter (1990). Best solution value over 100 trials.
7 Fiechter (1990). Worst solution value over 100 trials.
8 Fiechter (1990). SILICON GRAPHIC STATION (1.6 Mflops). Average running time over 100 trials.
9 Crowder and Padberg (1980).
10 Cerny (1985).
11 Grötschel, Jünger and Reinelt (1989).
12 Padberg and Rinaldi (1987).

ACKNOWLEDGEMENTS

This work was partially supported by the Canadian Natural Sciences and Engineering Research Council (grants OGP0038816 and OGPIN020). Thanks are also due to Louis Salvail for his valuable help with programming.

REFERENCES

Bartholdi, J.J. III, and L.K. Platzman. 1988. Heuristics Based on Spacefilling curves for Combinatorial Problems in Euclidean Space. Management Science. 34, 291-305.

Bland, R.G., and D.F. Shallcross. 1989. Large Traveling Salesman Problems Arising from Experiments in X-Ray Crystallography: A Preliminary Report on Computation. Operations Research Letters. 8, 125-128.

Bonomi, E., and J.-L. Lutton. 1984. The N-City Travelling Salesman Problem: Statistical Mechanics and the Metropolis Algorithm. SIAM Review. 26, 551-568.

Cerny, V. 1985. Thermodynamical Approach to the Travelling Salesman Problem: An Efficient Simulation Algorithm. Journal of Optimization Theory and Applications. 45, 41-51.

Chauny, F., A. Haurie, R. Loulou, and E. Wagneur. 1987. Sequencing Punch Operations in an FMS: A Three Dimensional Spacefilling Curve Approach. INFOR. 25, 26-45.

Crowder, H., and M.W. Padberg. 1980. Solving Large-Scale Symmetric Traveling Salesman Problems to Optimality. Management Science. 26, 495-509.

Daganzo, C.F. 1984. The Length of Tours in Zones of Different Shapes. Transportation Research B. 18, 135-145.

Fiechter, C.-N. 1990. A Parallel Tabu Search Algorithm for Large Scale Traveling Salesman Problems. Working Paper 90/1, Département de Mathématiques, École Polytechnique Fédérale de Lausanne, Switzerland.

Glover, F. 1977. Heuristic for Integer Programming Using Surrogate Constraints. Decision Sciences. 8, 156-166.

Golden, B.L., and C.C. Skiscim. 1986. Using Simulated Annealing to Solve Routing and Location Problems. Naval Research Logistics Quarterly. 33, 261-280.

Golden, B.L., and W.R. Stewart Jr. 1985. Empirical Analysis of Heuristics. In E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Schmoys, eds., The Traveling Salesman Problem. A Guided Tour of Combinatorial Optimization. Wiley, Chichester.

Grötschel, M., M. Jünger, and G. Reinelt. 1989. Via Minimization with Pin Preassignments and Layer Preference. Zeitschrift für Angewandte Mathematik und Mechanik. 69, 393-399.

Karp, R.M. 1977. Probabilistic Analysis of Partitioning Algorithms for the Traveling-Salesman Problem in the Plane. Mathematics of Operations Research. 2, 209-224.

Kirkpatrick, S., C.D. Gelatt Jr., and M.P. Vecchi. 1983. Optimization by Simulated Annealing. Science. 220, 671-680.

Knox, J. 1988. An Application of TABU Search to the Symmetric Traveling Salesman Problem. Ph.D. Thesis, Center for Applied Artificial Intelligence (CAAI), Graduate School of Business, University of Colorado.

Krolak, P.D., W. Felts, and G. Marble. 1971. A Man-Machine Approach Toward Solving the Traveling Salesman Problem. Communications of the Association for Computing Machinery. 14, 327-334.

Laporte, G., H. Mercure and Y. Nobert. 1986. An Exact Algorithm for the Asymmetrical Capacitated Vehicle Routing Problem. Networks. 16, 33-46.

Laporte, G., Y. Nobert, and M. Desrochers. 1985. Optimal Routing under Capacity and Distance Restrictions. Operations Research. 33, 1050-1073.

Lin, S. 1965. Computer Solutions of the Traveling Salesman Problem. Bell System Technical Journal. 44, 2245-2269.

Lin, S., and B.W. Kernighan. 1973. An Effective Heuristic Algorithm for the Traveling Salesman Problem. Operations Research. 21, 498-516.

Malek, M. 1988. Search Methods for Traveling Salesman Problems. Working Paper, University of Texas, Austin.

Metropolis, N., A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. 1953. Equation of State Calculations by Fast Computing Machines. Journal of Chemical Physics. 21, 1087-1091.

Norback, J.P., and R.F. Love. 1977. Geometric Approaches to Solving the Traveling Salesman Problem. Management Science. 23, 1208-1223.

Ong, H.L., and H.C. Huang. 1989. Asymptotic Expected Performance of Some TSP Heuristics: An Empirical Evaluation. European Journal of Operational Research. 43, 231-238.

Or, I. 1976. Traveling Salesman-Type Combinatorial Problems and their Relation to the Logistics of Regional Blood Banking. Doctoral Dissertation, Northwestern University, Evanston, Illinois.

Padberg, M.W., and G. Rinaldi. 1987. Optimization of a 532-City Symmetric Traveling Salesman Problem by Branch and Cut. Operations Research Letters. 6, 1-7.

Padberg, M.W., and G. Rinaldi. 1990. Facet Identification for the Symmetric Traveling Salesman Polytope. Mathematical Programming. 47, 219-257.

Preparata, F.P., and M.I. Shamos. 1985. Computational Geometry. Springer-Verlag, New York.

Reinelt, G. 1989. Fast Heuristics for Large Geometric Traveling Salesman Problems. Report No. 185, Institut für Mathematik, Universität Augsburg, Federal Republic of Germany.

Rosenkrantz, D., R. Sterns, and P. Lewis. 1977. An Analysis of Several Heuristics for the Traveling Salesman Problem. SIAM Journal on Computing. 6, 563-581.

Rossier, Y., M. Troyon, and T.M. Liebling. 1986. Probabilistic Exchange Algorithms and the Euclidian Traveling Salesman Problem. Operations Research Spektrum. 8, 151-164.

Soriano, P. 1989. Étude de nouvelles avenues de recherche proposées en optimisation combinatoire. Publication CRT-619, Centre de recherche sur les transports, Montréal.

Stewart, W.R. Jr. 1977. A Computationally Efficient Heuristic for the Traveling Salesman Problem. Proceedings of the 13th of S.E. TIMS. 75-85.

Stewart, W.R. Jr. 1987. Accelerated Branch Exchange Heuristics for Symmetric Traveling Salesman Problems. Networks. 17, 423-437.

Supowit, K., E. Reingold and D. Plaisted. 1983. The Travelling Salesman Problem and Minimum Matching in the Unit Square. SIAM Journal on Computing. 12, 145-156.

Troyon, M. 1988. Quelques heuristiques et résultats asymptotiques pour trois problèmes d'optimisation combinatoire. Thèse no. 754, École Polytechnique Fédérale de Lausanne, Switzerland.